#23

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: Nathan Abramson et al.

Serial No.: 09/209,015

Filed: December 10, 1998

For: SYSTEM AND METHOD FOR AUTOMATIC MAPPING OF HYPERTEXT INPUT FIELDS TO SOFTWARE COMPONENTS

Examiner: Rossi, Jeffrey A.

Art Unit: 2176

Commissioner for Patents
Washington, DC 20231

Duplicate Paper #17

APPELLANT'S BRIEF

Real Party of Interest

The Real Party of Interest is Art Technology Group, Inc.

Related Appeals and Interferences

It is believed that there are no other appeals or interferences which will directly affect or be directly affected by or have bearing on the board's decision in the pending appeal.

Status of Claims

Claims 1 – 17 have been finally rejected and are pending. As provided in a proposed amendment, claims 5 – 8 would be cancelled, leaving claims 1 - 4 and 9 - 17.

Status of Amendments

An amendment to the claims in response to a 35 U.S.C. 112, second paragraph, rejection is being submitted herewith and is therefore pending. That amendment would make a minor change to claim 1, cancel claims 5-8, and make a change to claim 16 to make it more similar to other pending claims. The enclosed Appendix reflects the changes made in the proposed Amendment After Final.

Summary of Invention

The invention is related to a method that includes rendering a hypertext document that includes emitting program code for mapping input field names in the hypertext document to software component properties when the hypertext document is rendered. The software component is a server-based component that uses data from the input field for processing. The method also includes providing the rendered hypertext document to a user, receiving from the user input field data that is entered in a named input field, and determining from the mapping, a software component property mapped to the named input field. The method further includes calling the software component for processing the input field data. The mapping is done so that the software component can process the input field data regardless of the spelling of the name of the input field in the hypertext document.

Referring to Fig. 2, this invention is further related to a web server system comprising a preprocessor (22), a name-space manager (24), and a data handler (26). The preprocessor generates program code to register mappings between hypertext input field names and software component properties, and to emit hypertext from tags. The software component is a server-based component that uses data from the input field for processing. The name-space manager registers the mappings. The data handler, responsive to input data submitted by a user with an input field name, uses the mappings in the name-space manager to associate the input field names with the appropriate component properties and calls the software component to process the input data. The mapping allows the software component to process the entered data regardless of the name of the input field in the hypertext documents.

Issues

Whether claims 1 – 3, 12 – 13, and 16 are anticipated by WO 98/46695 ("Apple"), and whether claims 4, 10, 11, 14, 15 and 17 would have been obvious over Apple.

Grouping of Claims

Group I : claims 1, 12, and 16

Group II : claims 2 – 4

Group III : claims 9 – 11, 13 – 15, and 17

Arguments

The Rejection over the Cited Reference

The Office Action rejected claims 1 – 3, 5 – 9, 2 – 13, and 16 under 35 U.S.C. 102(b) as being anticipated by WO 98/44695 (hereinafter referred to as "Apple"). Claims 10 and 14 are rejected under 35 U.S.C. 102(b) as being anticipated by or, in the alternative, under 35 U.S.C. 103(a) as obvious over Apple. Claims 4, 11, 15 and 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Apple.

Group I: Claims 1, 12, and 16

Apple is related to a method and apparatus for integrating applets running on a client with application logic for applications running on a server. An applet is a program that commences operation from a WWW (HTML) document and, in this method and apparatus, is used to replace the FORM element (page 12, lines 24 – 25). The applets are loaded from an "Applet Code" tag that specifies the applet code's location, and the applet's keys are bound to variables and methods within the server. These keys allow state in the applet to be associated with state in the server, and events in the applet to be associated with the invocation of methods in the server (pages 17, line 35 to page – 18, line 1).

An Applet Group Controller ("Controller") is responsible for managing the communication and data synchronization between the client and the server. Upon the invocation of an event, the Coordinator gathers the values for all the applet keys and compares the key values to the values stored in the dictionary. If the two values are different, the changed values are updated in the server, where they are then bound to variables and action logic is invoked. The changed values and keys are then returned to applets via the Coordinator, and the new values are displayed in the browser. Thus, Apple is related to a method and apparatus for updating and synchronizing the state of data between the client and the server.

In contrast, the recited invention is related to a method that includes rendering a hypertext document that includes emitting program code for mapping input field names in the hypertext document to software component properties when the hypertext document is rendered. The software component is a server-based component that uses data from the input field for processing. The method also includes providing the rendered hypertext document to a user,

receiving from the user input field data that is entered in a named input field, and determining from the mapping, a software component property mapped to the named input field. The method further includes calling the software component for processing the input field data. The mapping is done so that the software component can process the input field data regardless of the spelling of the name input field in the hypertext document.

Apple is therefore related to a method for integrating applets running on a client with the application logic for applications running on a server. The integration is specifically used for the synchronization of state and for the recognition of user actions in the browser. In contrast, claim 1 recites a method for emitting program code for mapping **input field names** in the hypertext document to **software component properties**. As stated in the specification, this mapping is used for the development and maintenance of an application, the re-use of software components, and avoids data type and/or naming errors.

Furthermore, this mapping is done so that the software component can process the input field data, regardless of the spelling of the name input field in the hypertext document. In contrast, Apple uses an "Applet Code" tag that provides the source of the applet code; as shown on page 13, lines 16 – 20, the value "next.wof.widgets.CalendarApplet.class" directs the browser to the location of the Calendar Applet code. If the source is misspelled or incorrectly provided, the browser will not be able to locate the Applet code or load an Applet. As mentioned above, an Applet and its keys are required for data synchronization. Therefore, the method and apparatus taught by Apple requires that the spelling of the applet source code be exact in order for the data synchronization to occur.

For at least these reasons, claim 1 is deemed allowable over the prior art. Furthermore, it follows that claims 12 and 16, which pertain to a web server system and method, respectively, should also be allowed for similar reasons.

Group II : Claims 2 - 4

Claims 2 and 3 were rejected under 35 U.S.C. 102(b) as being anticipated by Apple. Claim 4 was rejected under 35 U.S.C. 103(a) as being unpatentable over Apple.

Claims 2 - 4 are being treated as a group because they specifically pertain to hypertext form tags or a hypertext input form, and are patentable regardless of the patentability of claim 1. Claim 2 recites a method wherein the rendering includes emitting hypertext form tags with a current value of an input field pre-filled in. Claim 3 recites a method wherein mapping includes encoding the hypertext input form with a unique name and registering the name. Claim 4 recites a method wherein receiving from the user input field data entered in a named input field includes determining if the user submitted input field data is from a hypertext input form and bypassing input field processing if the determination cannot be made.

Apple, however, is related to a method and apparatus that uses applets instead of HTML forms: "To avoid the limitations present with the FORM element, the preferred embodiment of the present invention uses individual applets" (page 12, lines 23 – 26). Therefore, Apple teaches away from the invention recited in claims 2 and 3, because it specifically mentions using applets instead of hypertext form elements.

Furthermore, with respect to claim 4, it would not be obvious to one skilled in the art to modify Apple to bypass input filed processing, as is stated on page 11, paragraph 26, of the Office Action. As Apple uses applets instead of form elements, all key data is automatically collected upon the invocation of an action and is sent to the server. Therefore, one would not be motivated to modify Apple to determine if the user submitted input field data is from a hypertext input form or to bypass field input processing if the determination cannot be made.

For at least these reasons, claims 2 – 4 are separately patentable over Apple.

Group III: Claims 9 – 11, 13 – 15, and 17

Claims 9 – 11, 13 – 15, and 17 have been grouped together because they are related to input field data and input field names. These claims have additional limitations not taught by the cited reference, regardless of any disposition of claim 1, 12, or 16.

Claim 9 recites a method that further comprises converting the submitted input field data to a correct data type. The Applicants disagree with the arguments set forth in the Office Action, which refers to "HTML elements (including applets) are mapped to objects in an object-oriented environment" on page 14, lines 10 – 13 of Apple as anticipating the invention recited in claim 9.

Apple is related to different HTML elements being mapped to object classes so that that the object's method can be used to manipulate the HTML elements. Nowhere does this teach or suggest, however, converting input field data to a correct data type; the mapping of HTML elements to object classes is completely unrelated to data type or the conversion of data types.

By this reasoning, nowhere does Apple teach or suggest a data handler that converts the submitted input data to a correct data type (as is recited in claim 13). For at least these reasons, claim 9 and 13 are believed patentable over the prior art.

Claim 10 recites a method of determining from the mapping a software component property mapped to the named input field wherein the determining includes iteratively processing input names associated with a component property to determine if data associated with any of the input names has been entered. The Office Action states that "it would have been obvious to a person having ordinary skill in the art (PHOSITA) at the time of the invention to iterate Apple because it was suggested on page 21, lines 1 – 3, and in order to ensure synchronization as described by Apple" (page 10, paragraph 22).

As previously mentioned, Apple relates to a method and apparatus for synchronizing state between client and server in which, upon the invocation of an action, all applet keys are collected and compared against dictionary's values. Apple's method of simultaneously collecting all applet keys teaches away from iteratively processing input names to determine if data associated with any of the input names has been entered; Apple will collect the applet keys regardless of whether data is entered.

With respect to claim 11, the Office Action states that "it would have been obvious to PHOSITA at the time of the invention to process input fields in a prioritized order in order to give preference to the executions of certain actions over another." In order to assign priority levels in Apple, however, the system would first have to compare the values of the applet keys to the values in the dictionary, and then determine which values were different. The changed values would be assigned a higher priority than unchanged values. Therefore, in Apple, priority levels would have to be derived based upon the comparison of key values. Nowhere does Apple teach or suggest, however, initially processing in order of priority that are **stored** with the mapping of the input fields.

Claim 15 recites a name-space manger that includes a table for mapping a form to input fields, and for mapping input fields to a priority that determines the order in which the data handler processes the input field mappings. Claim 17 recites a method wherein the storing (of input field names) includes storing a priority for each input name. For at least the aforementioned reasons, claims 15 and 17 are patentable over the prior art of record; nowhere does Apple teach or suggest storing priority for each input name.

Claim 14 recites a name-space manager that includes a table for mapping a form to input fields, and for mapping input fields to a component property. As previously mentioned, Apple is related to a method and apparatus for comparing applet key values to values in a dictionary. Furthermore, Apple teaches away from using a form and input fields and does not teach or suggest mapping input fields to a priority . For at least there reasons, claim 14 is patentable over the prior art.

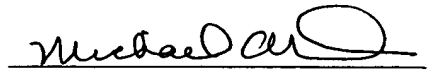
Conclusion

All claims should now be in condition for allowance and accordingly a notice of allowance is respectfully requested.

The Commissioner is hereby authorized to charge any fees now required to maintain the pendency of the application, to Deposit Account No. 08-0219.

Respectfully submitted,

Date: February 7, 2003



Michael A. Diener
Reg. No. 37,122
Attorney for Applicant

Hale and Dorr LLP
60 State Street
Boston, MA 02109
(617) 526-6454

APPENDIX

1. A method comprising:

rendering a hypertext document including emitting program code for mapping input field names in the hypertext document to software component properties when the hypertext document is rendered, a software component being a server-based component that uses data from the input field for processing;

providing the rendered hypertext document to a user;

receiving from the user input field data entered in a named input field;

determining from the mapping a software component property mapped to the named input field; and

calling the software component for processing the input field data, the mapping being done such that the software component can process the input field data regardless of the spelling of the name of the input field in the hypertext document.

2. The method of claim 1, wherein the rendering includes emitting hypertext form tags with a current value of an input field pre-filled in.

3. The method of claim 1, wherein the mapping includes encoding the hypertext input form with a unique name and registering the name.

4. The method of claim 3, wherein the receiving includes determining if the user submitted input field data is from a hypertext input form and bypassing input field processing if the determination cannot be made.

9. The method of claim 1, further comprising converting the submitted input field data to a correct data type.

10. The method of claim 1, wherein the determining includes iteratively processing input names associated with a component property to determine if data associated with any of the input names has been entered.

11. The method of claim 10, wherein the determining includes processing in order of priority stored with the mapping of the input field names.

12. A web server system comprising:

a preprocessor for generating program code to register mappings between hypertext input field names and software component properties and to emit hypertext form tags, the software component being a server-based component that uses data from the input field for processing;

a name-space manager for registering the mappings; and

a data handler, responsive to input data submitted by a user with an input field name, for using the mappings in the name-space manager to associate the input field names with the appropriate component properties, and for calling the software component to process the input data, the mapping allowing the software component to process the entered data regardless of the name of the input field in the hypertext document.

13. The system of claim 12, wherein the data handler converts the submitted input data to a correct data type.

14. The system of claim 12, wherein the name-space manager includes a table for mapping a form to input fields, and for mapping input fields to a component property.

15. The system of claim 12, wherein the name-space manager includes a table for mapping a form to input fields, and for mapping input fields to a priority that determines the order in which the data handler processes the input field mappings.

16. A method comprising processing a hypertext document by identifying and storing input field names before sending the document to a user, and in response to receiving input data with input field names from the user, determining appropriate software component methods for processing the input data by looking to the stored input field names, the input names for the input data being determined from the document regardless of the naming of the input within the hypertext document.

17. The method of claim 16, wherein the storing includes storing a priority for each input name.